

Seminarski rad

BAZA PODATAKA ZA TURISTIČKI KAMP

Moderni sustavi baza podataka

Ana Koturić

II. godina, preddiplomski studij matematike i računarstva

Sadržaj

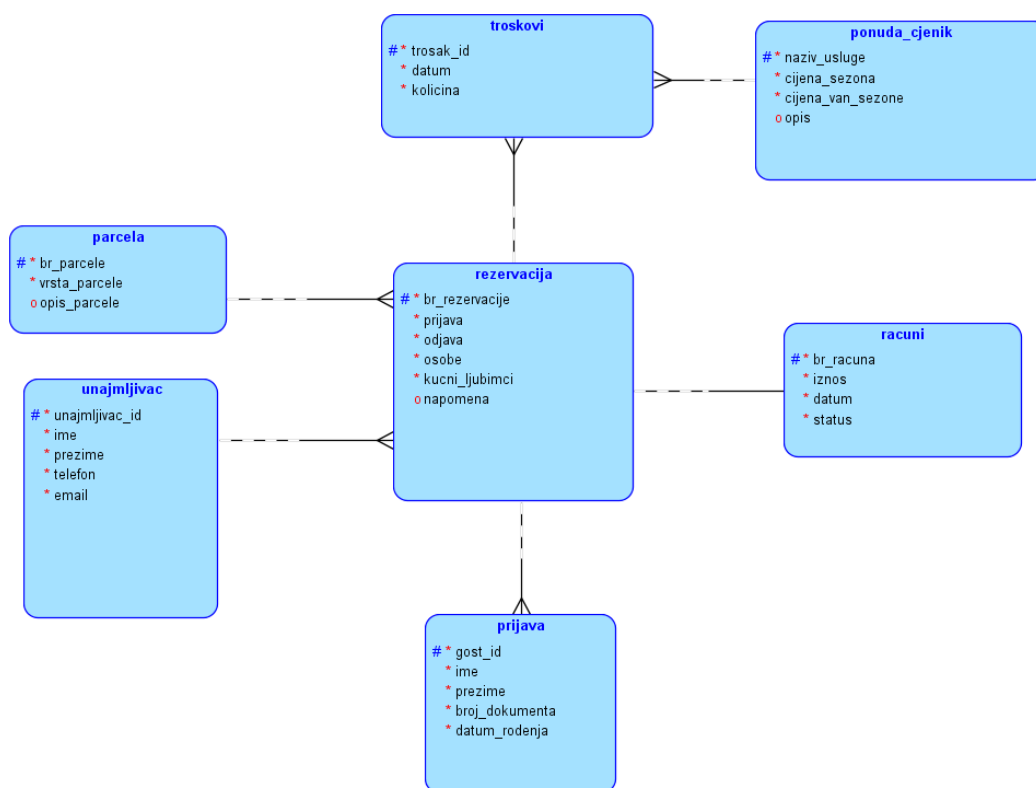
UVOD.....	3
MEV i relacijski model.....	3
Entiteti i atributi.....	3
Veze između entiteta	4
Relacijska model	5
Procedure, okidači, indexi i upiti.....	6
Procedura unos_azuriranje_parcele.....	6
Procedura dostupnost_parcela.....	8
Procedura rezerviranje	9
Procedura gost_prijava	11
Procedura unos_azuriranje_ponude	12
Procedura unos_troskovi	14
Procedura izdavanje_racuna.....	15
Procedura placanje_racuna	18
Upiti.....	19
Indexi.....	22

UVOD

Cilj ovog projekta je bio napraviti bazu podataka za turistički kamp koja bi zaposlenicima kampa olakšala praćenje rezervacija, troškova, gostiju i izdavanje računa. Ovaj seminar će biti kratki vodič za korištenje baze podataka za turistički kamp.

MEV i relacijski model

Na sljedećoj slici nalazi se Model entita i veza (MEV).



Entiteti i atributi

MEV se sastoji od sedam entiteta i njihovih atributa, a to su:

- **parcela** -> entitet za evidentiranje svih parcela koje se nalaze u kampu
 - o br_parcele: number(5) PRIMARY KEY, obavezan atribut
 - o vrsta_parcele: varchar(20), obavezan atribut ->može biti 'sator', 'kamp kucica' ili 'kamp prikolica'
 - o opis_parcele: varchar(100), opcionalan atribut

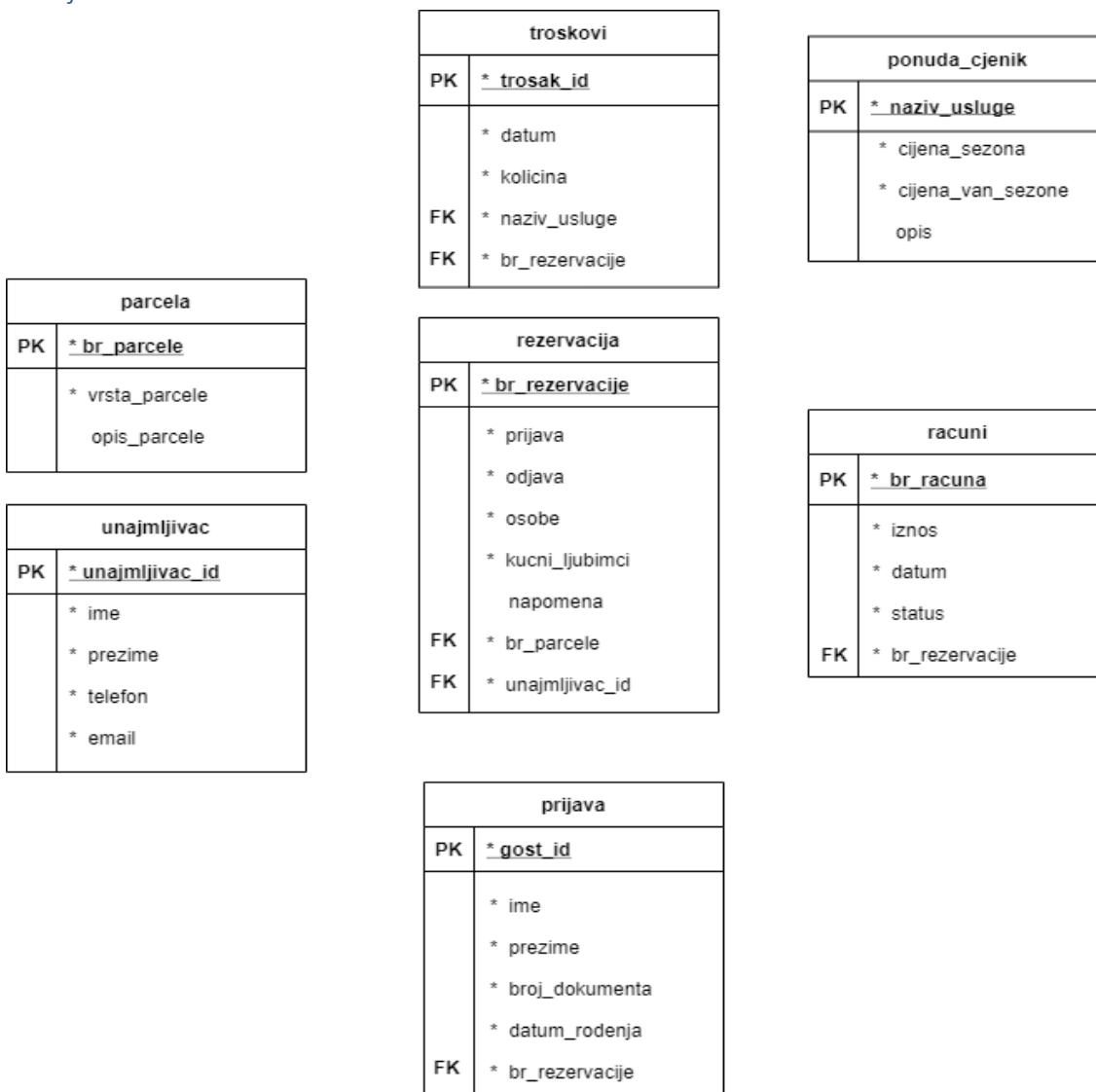
- **unajmljivac** -> entitet u koji pohranjujemo podatke o osobi koja je rezervirala parcelu, a ta osoba ne mora nužno biti i gost kampa.
 - unajmljivac_id: varchar(20) PRIMARY KEY, obavezan atribut
 - ime: varchar(30), obavezan atribut
 - prezime: varchar(30), obavezan atribut
 - telefon: varchar(20), obavezan atribut
 - email: varchar(20), obavezan atribut
 - ime: varchar(30), obavezan atribut
- **rezervacija** -> entite pomoću kojeg evidentiramo sve rezervacije i najbitnije informacije vezane za njih.
 - br_rezervacije: varchar(20) PRIMARY KEY, obavezan atribut
 - prijava: date, obavezan atribut
 - odjava: date, obavezan atribut
 - osobe: number(2), obavezan atribut -> ovaj argument se odnosi na broj osoba, tj. gostiju
 - kucni_ljubimci: number(1), obavezan atribut -> ovaj argument se odnosi na broj kucnih ljubimaca
 - napomena: varchar(100), opcionalan atribut
- **prijava** -> kako je potrebno imati informacije o svim gostima koji su stigli u kamp, pomoću ovog entiteta ćemo evidentirati sve pristigle goste.
 - gost_id: varchar(20) PRIMARY KEY, obavezan atribut
 - ime: varchar(30), obavezan atribut
 - prezime: varchar(30), obavezan atribut
 - broj_dokumenta: varchar(30), obavezan atribut
 - datum_rođenja: date, obavezan atribut
- **ponuda_cjenik** -> entite u koji ćemo spremiti sve usluge koje nudi kamp i njihove cijene, kako bi lakše izračunali cijenu računa.
 - naziv_usluge: varchar(30) PRIMARY KEY, obavezan atribut
 - cijena_sezona: number(5,2), obavezan atribut
 - cijena_van_sezone: number(5,2), obavezan atribut
 - opis: varchar(100), opcionalan atribut
- **troškovi** -> entitet u kojem ćemo voditi evidenciju o troškovima vezanim za svaku rezervaciju.
 - trosak_id: varchar(20) PRIMARY KEY, obavezan atribut
 - datum: date, obavezan atribut
 - kolicina: number(3), obavezan atribut -> količina će se odnositi, ovisno o trošku, na broj dana, broj osoba, broj korištenja ili broj sati.
- **racuni** -> entitet u koji ćemo spremiti sve izdane i/ili plaćene račune.
 - br_racuna: varchar(20) PRIMARY KEY, obavezan atribut
 - iznos: number(7,2), obavezan atribut
 - datum: date, obavezan atribut
 - status: varchar(20), obavezan atribut -> status može biti 'izdan' ili 'placen'.

Veze između entiteta

- **veza parcela-rezervacija:** Veza jedan naprama više. Parcela može i ne mora imati jednu ili više rezervacija, dok rezervacija mora imati točno jednu parcelu. Zbog te veze, rezervacija nasljeđuje primarni ključ br_parcele od entiteta parcela i on postaje strani ključ rezervacije.
- **veza unajmljivac-rezervacija:** Veza jedan naprama više. Unajmljivač može i ne mora imati jednu ili više rezervacija, dok rezervacija mora imati točno jednog unajmljivača. Primarni ključ unajmljivac_id iz unajmljivac, postaje strani ključ u rezervacija.

- **veza rezervacija-prijava:** Veza jedan prema više. Jedna rezervacija može imati jednu, nijednu ili više prijava, dok prijava mora imati točno jednu rezervaciju. Entitet prijava nasljeđuje primarni ključ br_rezervacije iz rezervacije, koji mu postaje strani ključ.
- **veza rezervacija-troskovi:** Veza jedan prema više. Rezervacija može i ne mora imati jedan ili više troškova, ali trošak mora imati točno jednu rezervaciju. Primarni ključ br_rezervacije iz rezervacija postaje strani ključ entiteta troskovi.
- **Veza troskovi-ponuda_cjenik:** Veza jedan naprama više. Trošak ima točno jednu uslugu iz entiteta ponuda_cjenik, dok usluga ne mora biti vezana niti za jedan trošak. Entite troskovi nasljeđuje primarni ključ naziv_usluge iz ponuda_cjenik i naziv_usluge postaje strani ključ entiteta troskovi.
- **veza rezervacija-racuni:** Veza jedan naprama jedan. Rezervacija može imati jedan račun, ali račun mora imati jednu rezervaciju. Entitet racun nasljeđuje primarni ključ br_rezervacije iz rezervacija i on mu postaje strani ključ.

Relacijska model



Na prethodnoj slici možemo vidjeti relacijski model za bazu podataka turistički kamp. Relacijski model pretvara MEV u tablice, kako bi lakše mogli kreirati bazu podataka. U relacijskom modelu svaki entitet je jedna tablica i ta tablica sadrži sve argumente koji su navedeni u entitetu u MEV-u, ali uz njih još ovisno o vezama između entiteta dobiva neke strane ključeve. Tako je od entiteta *rezervacija* nastala tablica rezervacija, koja uz argumente iz entiteta ima još i argumente *br_parcele* i *unajmljivac_id* koji su strani ključevi.

Procedure, okidači, indexi i upiti

Procedura unos_azuriranje_parcela

Procedura *unos_azuriranje_parcela* je procedura pomoću koje unosimo nove parcele u bazu i ažuriramo stare.

```
CREATE PROCEDURE unos_azuriranje_parcela(
    p_br_parcele IN parcela.br_parcele%TYPE,
    p_vrsta_parcele IN parcela.vrsta_parcele%TYPE,
    p_opis_parcele IN parcela.opis_parcele%TYPE
)
AS
    p_count number;
BEGIN
    SELECT COUNT(*)
    INTO p_count
    FROM parcela
    WHERE br_parcele = p_br_parcele;

    IF p_count = 0 THEN
        IF p_vrsta_parcele IS NOT NULL THEN
            INSERT INTO parcela(br_parcele, vrsta_parcele, opis_parcele) VALUES(p_br_parcele, p_vrsta_parcele, p_opis_parcele);
            COMMIT;
        ELSE
            DBMS_OUTPUT.PUT_LINE('Niste unijeli obavezne argumente za unos nove parcele.');
        END IF;
    ELSIF p_count = 1 THEN
        IF p_vrsta_parcele IS NOT NULL AND p_opis_parcele IS NOT NULL THEN
            UPDATE parcela
            SET parcela.vrsta_parcele = p_vrsta_parcele, parcela.opis_parcele = p_opis_parcele
            WHERE br_parcele = p_br_parcele;
        ELSIF p_vrsta_parcele IS NOT NULL AND p_opis_parcele IS NULL THEN
            UPDATE parcela
            SET parcela.vrsta_parcele = p_vrsta_parcele
            WHERE br_parcele = p_br_parcele;
        ELSIF p_vrsta_parcele IS NULL AND p_opis_parcele IS NOT NULL THEN
            UPDATE parcela
            SET parcela.opis_parcele = p_opis_parcele
            WHERE br_parcele = p_br_parcele;
        ELSE
            DBMS_OUTPUT.PUT_LINE('Pogreska.');
        END IF;
    COMMIT;
    END IF;
END unos_azuriranje_parcela;
/
```

Za unos nove parcele potrebno je pokrenuti proceduru koja se nalazi na sljedećoj slici i unijeti broj parcele, vrstu parcele, a nju određujemo prema tome je li namjenjena za šator, kamp kućicu ili kamp prikolicu, pa je ovisno o tome potrebno unijeti jedan od sljedećih pojmova: *sator*, *kamp kucica*, *kamp prikolica*, i po želji opis parcele, ukoliko ne želite unijeti opis potrebno je napisati *null*.

Unosimo parcelu broj 11, namjenjenu za šator s opisom 'Mala parcela za šator, pored masline.'

```
CALL unos_azuriranje_parcela(11, 'sator', 'Mala parcela za sator, pored masline.');
```

Ako želite ažurirati parcelu pokrenut ćete proceduru *unos_azuriranje_parcela* i unijeti broj parcele, te argumente koje želite ažurirati, a za one koje ne želite ažurirati upišite *null*.

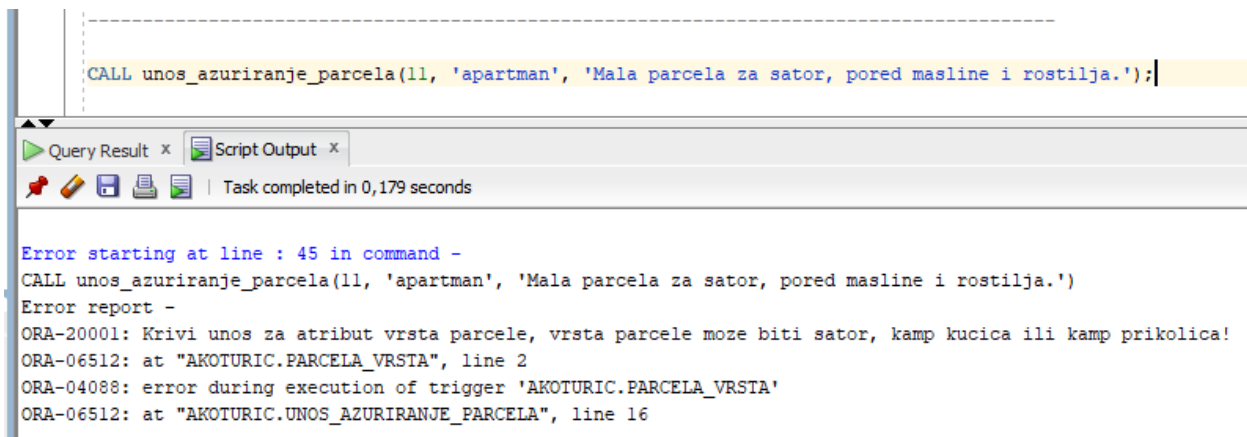
Ažuriramo opis parcele broj 11.

```
CALL unos_azuriranje_parcela(11, null, 'Mala parcela za sator, pored masline.');
```

Ukoliko za vrstu parcele unesete neku riječ koja nije 'sator', 'kamp kucica' ili 'kamp prikolica', aktivirat će se okidač sa sljedeće slike koji će spriječiti izvršavanje naredbe.

```
CREATE TRIGGER parcela_vrsta
BEFORE INSERT ON parcela
FOR EACH ROW WHEN
  (NEW.vrsta_parcele != 'sator' AND NEW.vrsta_parcele != 'kamp kucica' AND NEW.vrsta_parcele != 'kamp prikolica')
BEGIN
  RAISE_APPLICATION_ERROR(-20001, 'Krivi unos za atribut vrsta parcele, vrsta parcele moze biti sator, kamp kucica ili kamp prikolica!');
END parcela_vrsta;
```

I dobit ćete sljedeće upozorenje.



```
CALL unos_azuriranje_parcela(11, 'apartman', 'Mala parcela za sator, pored masline i rostilja.');
```

Query Result x Script Output x

Task completed in 0,179 seconds

Error starting at line : 45 in command -
CALL unos_azuriranje_parcela(11, 'apartman', 'Mala parcela za sator, pored masline i rostilja.')

Error report -
ORA-20001: Krivi unos za atribut vrsta parcele, vrsta parcele moze biti sator, kamp kucica ili kamp prikolica!
ORA-06512: at "AKOTURIC.PARCELA_VRSTA", line 2
ORA-04088: error during execution of trigger 'AKOTURIC.PARCELA_VRSTA'
ORA-06512: at "AKOTURIC.UNOS_AZURIRANJE_PARCELA", line 16

Procedura dostupnost_parcela

Pomoću procedure *dostupnost_parcela* ćemo lagano saznati koje parcele su dostupne za dano razdoblje i vrstu parcele.

```
CREATE PROCEDURE dostupnost_parcela(
    p_vrsta IN parcela.vrsta_parcele%TYPE,
    p_datum_prijave IN rezervacija.prijava%TYPE,
    p_datum_odjave IN rezervacija.prijava%TYPE
)
AS
BEGIN
    IF (p_vrsta = 'sator' OR p_vrsta = 'kamp kucica' OR p_vrsta = 'kamp prikolica') AND p_datum_prijave < p_datum_odjave THEN

        DECLARE
            c_br_parcele parcela.br_parcele%TYPE;
            c_vrsta_parcele parcela.vrsta_parcele%TYPE;
            c_opis_parcele parcela.opis_parcele%TYPE;

            CURSOR printaj_parcele IS
                SELECT *
                FROM parcela pa
                WHERE vrsta_parcele = p_vrsta AND br_parcele NOT IN (
                    SELECT br_parcele
                    FROM rezervacija
                    WHERE br_parcele = pa.br_parcele AND p_datum_prijave < odjava AND p_datum_odjave > prijava);

        BEGIN
            DBMS_OUTPUT.PUT_LINE('Dostupne parcele su: ');
            OPEN printaj_parcele;

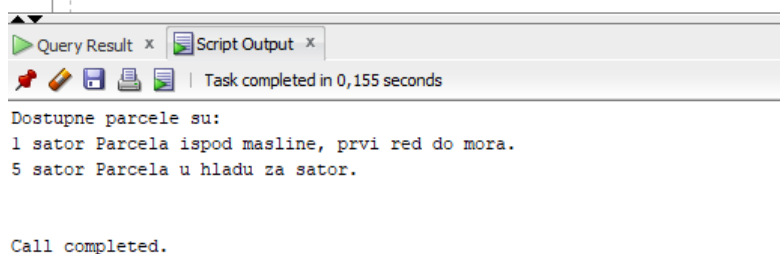
            LOOP
                FETCH printaj_parcele
                INTO c_br_parcele, c_vrsta_parcele, c_opis_parcele;
                EXIT WHEN printaj_parcele%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE(c_br_parcele || ' ' || c_vrsta_parcele || ' ' || c_opis_parcele);
            END LOOP;

            CLOSE printaj_parcele;
        END;

    ELSIF p_datum_prijave > p_datum_odjave THEN
        DBMS_OUTPUT.PUT_LINE('Datum prijave mora biti prije datuma odjave!');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Pogresan unos za vrstu. Vrsta moze biti sator, kamp kucica ili kamp prikolica.');
```

Za pokretanje procedure potrebno je unijeti vrstu parcele, datum dolaska i datum odlaska.

```
CALL dostupnost_parcela('sator', '09-07-2020', '20-07-2020');
```



Ako zamijenite datum dolaska i datum odlaska dobit ćete sljedeću poruku:

```
CALL dostupnost_parcela('sator', '20-07-2020', '09-07-2020');
```

Task completed in 0,49 seconds

Datum prijave mora biti prije datuma odjave!

Call completed.

Također ako unesete krivu riječ za vrstu parcele dobit ćete poruku:

```
CALL dostupnost_parcela('apartman', '09-07-2020', '20-07-2020');
```

Task completed in 0,344 seconds

Pogresan unos za vrstu. Vrsta može biti sator, kamp kucica ili kamp prikolica.

Call completed.

Procedura rezerviranje

Procedura *rezerviranje* nam služi za unos rezervacije i unajmljivača. Potrebno je unijeti osobne podatke o unajmljivaču te njegov kontakt, kako bi imali potrebne informacije u slučaju da dođe do promjene vezano za rezervaciju. Također unosimo osnovne informacije o rezervaciji, datum prijave i datum odjave, broj parcele koja se rezervira, broj osoba, broj kućnih ljubimaca i po potrebi napomenu. Ako ne želite unijeti napomenu, umjesto nje upisujete *null*.

```
CALL rezerviranje('Dora', 'Doric', '0951423666', 'dorad@gmail.com', 1, '09-07-2020', '20-07-2020', 3, 0, null);
```

Gornjom procedurom unijeli smo unajmljivača Dora Doric u tablicu *unajmljivaci* i također rezervaciju za parcelu broj 1 u tablicu *rezervacija*.

```
SELECT *
FROM unajmljivac
WHERE ime = 'Dora' AND prezime='Doric';
```

Script Output x Query Result x

All Rows Fetched: 1 in 0,045 seconds

UNAJMLJIVAC_ID	IME	PREZIME	TELEFON	EMAIL
1 U41	Dora	Doric	0951423666	dorad@gmail.com

```
SELECT *
FROM rezervacija
WHERE unajmljivac_id='U41';
```

Script Output x Query Result x

All Rows Fetched: 1 in 0,045 seconds

BR_REZERVACIJE	PRIJAVA	ODJAVA	OSOBE	KUCNI_LJUBIMCI	NAPOMENA	BR_PARCELE	UNAJMLJIVAC_ID
1 R10040	09.07.20	20.07.20	3	0 (null)		1 U41	

```

CREATE PROCEDURE rezerviranje(
    p_ime IN unajmljivac.ime%TYPE,
    p_prezime IN unajmljivac.prezime%TYPE,
    p_telefon IN unajmljivac.telefon%TYPE,
    p_email IN unajmljivac.email%TYPE,
    p_br_parcele IN parcela.br_parcele%TYPE,
    p_prijava IN rezervacija.prijava%TYPE,
    p_odjava IN rezervacija.odjava%TYPE,
    p_osobe IN rezervacija.osobe%TYPE,
    p_kucni_ljubimci IN rezervacija.kucni_ljubimci%TYPE,
    p_napomena IN rezervacija.napomena%TYPE
)
AS
    p_unajmljivac_id varchar(20);
    p_count number;
    p_count_unajmljivac number;
    p_br_rez varchar(20);
BEGIN
    IF p_prijava < p_odjava THEN
        SELECT COUNT(*)
        INTO p_count
        FROM ( SELECT br_parcele FROM parcela pa
              WHERE br_parcele NOT IN
                (SELECT br_parcele FROM rezervacija r
                 WHERE br_parcele = pa.br_parcele AND p_prijava < r.odjava AND p_odjava > r.prijava))
        WHERE br_parcele = p_br_parcele;

        IF p_count = 1 THEN
            SELECT COUNT(*)
            INTO p_count_unajmljivac
            FROM unajmljivac
            WHERE ime = p_ime AND prezime = p_prezime AND telefon = p_telefon AND email = p_email;

            IF p_count_unajmljivac = 1 THEN
                SELECT unajmljivac_id
                INTO p_unajmljivac_id
                FROM unajmljivac
                WHERE ime = p_ime AND prezime = p_prezime AND telefon = p_telefon AND email = p_email;
            ELSE
                SELECT CONCAT('U', brojevi_u.nextval)
                INTO p_unajmljivac_id
                FROM DUAL;
                INSERT INTO unajmljivac(unajmljivac_id, ime, prezime, telefon, email)
                VALUES(p_unajmljivac_id, p_ime, p_prezime, p_telefon, p_email);
                COMMIT;
            END IF;

            SELECT CONCAT('R', brojevi_r.nextval)
            INTO p_br_rez
            FROM DUAL;

            INSERT INTO rezervacija(br_rezervacije, prijava, odjava, osobe, kucni_ljubimci, napomena, br_parcele, unajmljivac_id)
            VALUES(p_br_rez, p_prijava, p_odjava, p_osobe, p_kucni_ljubimci, p_napomena, p_br_parcele, p_unajmljivac_id);
            COMMIT;
        ELSE
            DBMS_OUTPUT.PUT_LINE('Parcela ' || p_br_parcele || ' nije dostupna u traženom razdoblju');
        END IF;
    ELSE
        DBMS_OUTPUT.PUT_LINE('Datum odjave mora biti nakon datuma prijave!');
    END IF;
END rezerviranje;
/

```

Procedura gost_prijava

Nakon što je unajmljivač rezervirao parcelu, dolazi datum prijave kada stižu gosti u kamp i potrebno ih je prijaviti kako bi imali potrebne podatke o njima zbog evidencije. Procedura gost_prijava omogućuje nam jednostavan unos gostiju u tablicu *prijava*.

```
CREATE PROCEDURE gost_prijava(
    p_ime IN prijava.ime%TYPE,
    p_prezime IN prijava.prezime%TYPE,
    p_broj_dokumenta IN prijava.broj_dokumenta%TYPE,
    p_datum_rodenja IN prijava.datum_rodenja%TYPE,
    p_br_rezervacije IN prijava.br_rezervacije%TYPE )
AS
    p_count_rezervacija number;
    p_count_osobe number;
    p_br_osoba number;
    p_g_id varchar(20);
BEGIN
    SELECT COUNT(*)
    INTO p_count_rezervacija
    FROM rezervacija
    WHERE br_rezervacije = p_br_rezervacije;

    IF p_count_rezervacija = 1 THEN
        SELECT COUNT(*)
        INTO p_count_osobe
        FROM prijava
        WHERE br_rezervacije = p_br_rezervacije;

        SELECT osobe
        INTO p_br_osoba
        FROM rezervacija
        WHERE br_rezervacije = p_br_rezervacije;

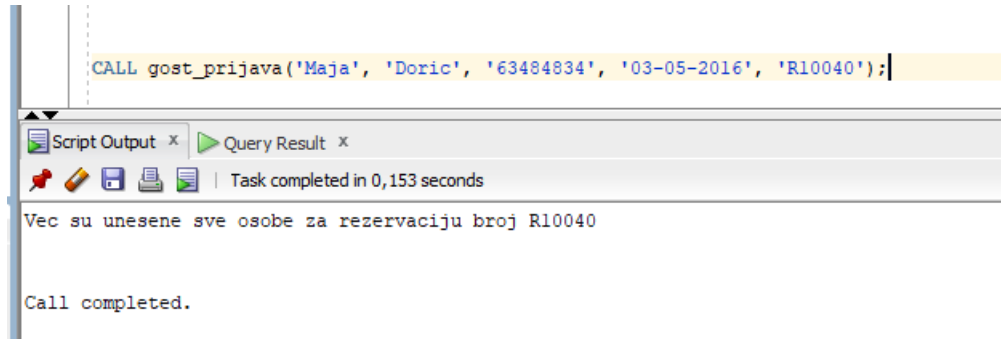
        IF p_count_osobe < p_br_osoba THEN
            SELECT CONCAT('G', brojevi_g.nextval)
            INTO p_g_id
            FROM DUAL;

            INSERT INTO prijava(gost_id, ime, prezime, broj_dokumenta, datum_rodenja, br_rezervacije)
            VALUES(p_g_id, p_ime, p_prezime, p_broj_dokumenta, p_datum_rodenja, p_br_rezervacije);
            COMMIT;
        ELSE
            DBMS_OUTPUT.PUT_LINE('Vec su unesene sve osobe za rezervaciju broj ' || p_br_rezervacije);
        END IF;
    ELSE
        DBMS_OUTPUT.PUT_LINE('Ne postoji rezervacija s brojem ' || p_br_rezervacije);
    END IF;
END gost_prijava;
```

Za pokretanje procedure *gost_prijava* potrebno je upisati ime, prezime, broj dokumenta, datum rođenja i broj rezervacije. Dora Doric iz prethodnog primjera rezervirala je mjesto za 3 osobe, došao je 9.7.2020. i gosti su stigli, pa ih unosimo u bazu.

```
CALL gost_prijava('Dora', 'Doric', '4352789', '14-08-1993', 'R10040');
CALL gost_prijava('Marko', 'Doric', '8352368', '19-03-1992', 'R10040');
CALL gost_prijava('Josipa', 'Doric', '4628482', '03-04-2016', 'R10040');
```

Ako pokušamo unijeti 4. osobu, dobit ćemo poruku da su već unesene sve osobe ta rezervaciju, jer je u rezervaciji navedeno da dolaze 3 osobe.



Ako pokrenemo upit za ispis svih prijavljenih osoba koje su vezane za rezervaciju 'R10040' tj rezervaciju Dore Doric dobit ćemo sljedeći rezultat:

```
SELECT *
FROM prijava
WHERE br_rezervacije = 'R10040';
```

The screenshot shows a query result window with the following table:

	GOST_ID	IME	PREZIME	BROJ_DOKUMENTA	DATUM_RODENJA	BR_REZERVACIJE
1	G162	Dora	Doric	4352789	14.08.93	R10040
2	G163	Marko	Doric	8352368	19.03.92	R10040
3	G164	Josipa	Doric	4628482	03.04.16	R10040

Možemo primjetiti da svaki gost ima svoj ID, a njega nismo unosili prilikom pokretanja procedure, *gost_id* se generira automatski tako što konkatimiramo slovo 'G' sa brojem koji se generira pomoću niza *brojevi_g*. Na taj način smo također generirali i broj rezervacije i ID unajmljivača.

```
--niz brojeva za ID gasta
CREATE SEQUENCE brojevi_g
START WITH 100
INCREMENT BY 1;
```

Procedura unos_azuriranje_ponude

Procedura *unos_azuriranje_ponude* je procedura pomoću koje unosimo sve usluge, odnosno sve ono što kamp nudi gostima, u tablicu *ponuda_cjenik*. U toj tablici je zabilježena cijena u sezoni i cijena izvan sezone za svaku uslugu i uz pomoć nje ćemo moći lakše izračunati račun.

```
CREATE PROCEDURE unos_azuriranje_ponude(
    p_naziv_usluge IN ponuda_cjenik.naziv_usluge%TYPE,
    p_cijena_sezona IN ponuda_cjenik.naziv_usluge%TYPE,
    p_cijena_van_sezone IN ponuda_cjenik.naziv_usluge%TYPE,
    p_opis IN ponuda_cjenik.naziv_usluge%TYPE )
AS
    p_count number;
BEGIN
    SELECT COUNT(*)
    INTO p_count
    FROM ponuda_cjenik
    WHERE naziv_usluge = p_naziv_usluge;

    IF p_naziv_usluge IS NOT NULL THEN
        IF p_count = 0 THEN
            IF p_cijena_sezona IS NOT NULL AND p_cijena_van_sezone IS NOT NULL THEN
                INSERT INTO ponuda_cjenik(naziv_usluge, cijena_sezona, cijena_van_sezone, opis)
                VALUES(p_naziv_usluge, p_cijena_sezona, p_cijena_van_sezone, p_opis);
                COMMIT;
            ELSE
                DBMS_OUTPUT.PUT_LINE('Za unos nove usluge potrebno je unijeti cijenu u sezoni i cijenu van sezone.');
            END IF;
        ELSE
            IF p_cijena_sezona IS NOT NULL THEN
                UPDATE ponuda_cjenik
                SET cijena_sezona = p_cijena_sezona
                WHERE naziv_usluge = p_naziv_usluge;
            END IF;
            IF p_cijena_van_sezone IS NOT NULL THEN
                UPDATE ponuda_cjenik
                SET cijena_van_sezone = p_cijena_van_sezone
                WHERE naziv_usluge = p_naziv_usluge;
            END IF;
            IF p_opis IS NOT NULL THEN
                UPDATE ponuda_cjenik
                SET opis = p_opis
                WHERE naziv_usluge = p_naziv_usluge;
            END IF;
            COMMIT;
        END IF;
    ELSE
        DBMS_OUTPUT.PUT_LINE('Potrebno je unijet naziv usluge kako bi unijeli novu uslugu ili kako bi azurirali postojećeg.');
    END IF;
END unos_azuriranje_ponude;
```

Kako bismo unijeli novu uslugu pomoću procedure *unos_azuriranje_ponude* potrebno je unijeti naziv usluge, cijenu u sezoni, cijenu izvan sezone i kratki opis ako je potrebno, u slučaju da ne želite unijeti opis, trebate napisati *null*.

```
CALL unos_azuriranje_ponude('lezaljka', 25, 20, 'Lezaljka za suncanje, cijena po danu');
```

Ako zatražimo ispis iz tablice *usluga_cjenik* na sljedeći način, dobit ćemo:

```
SELECT *
FROM ponuda_cjenik
WHERE naziv_usluge='lezaljka';
```

NAZIV_USLUGE	CIJENA_SEZONA	CIJENA_VAN_SEZONE	OPIS
1 lezaljka	25	20	Lezaljka za suncanje, cijena po danu

Ako odlučimo žurirati cijenu ležaljke, opet ćemo pozvati proceduru *unos_azuriranje_ponude*.

```
CALL unos_azuriranje_ponude('lezaljka', 35, null, null);

SELECT *
FROM ponuda_cjenik
WHERE naziv_usluge='lezaljka';
```

NAZIV_USLUGE	CIJENA_SEZONA	CIJENA_VAN_SEZONE	OPIS
1 lezaljka	35	20	Lezaljka za suncanje, cijena po danu

U procedure smo unijeli samo naziv usluge i cijenu u sezoni, a za ostale argumente *null* jer smo htjeli samo ažurirati cijenu u sezoni.

Procedura unos_troskovi

Procedura *unos_troskovi* služi nam za unos svih troškova koje nemožemo izvući iz rezervacije. Svaki trošak ima svoj ID, koji generiramo kao ID za gosta.

Prilikom unosa troškova potrebno je unijeti datum troška, naziv usluge kako bismo prilikom izračuna računa mogli povući cijenu iz *ponuda_cjenik*, količinu i broj rezervacije. Količina opisuje broj dana, sati, korištenja ili slično, ovisno o tome o kojoj se usluzi radi, a u opisu usluge je opisano naplaćuje li se usluga po danu, satu, korištenju i slično.

Vratit ćemo se na rezervaciju Dore Doric, gosti su stigli u kamp i kreću koristiti usluge kampa. Stigli su autom, a u kampu se parking naplaćuje posebno, pa je potrebno unijeti taj trošak.

```
CALL unos_troskovi('09-07-2020', 'parking auto', 11, 'R10040');
```

U gornjem pozivu procedure, treći argument je količina, gosti ostaju 11 dana, a parking se naplaćuje po danu, pa je zato količina 11.

<pre> SELECT * FROM troskovi WHERE br_rezervacije = 'R10040'; </pre>					
<div> <div>Script Output x</div> <div>Query Result x</div> </div> <div> <div> </div> <div> <div>SQL</div> <div>All Rows Fetched: 1 in 0,069 seconds</div> </div> </div>					
TROSAK_ID	DATUM	NAZIV_USLUGE	KOLICINA	BR_REZERVACIJE	
1 T61	09.07.20	parking auto	11	R10040	

```

--niz borjeva za ID troska
CREATE SEQUENCE brojevi_t
  START WITH 1
  INCREMENT BY 1;

CREATE PROCEDURE unos_troskovi(
  p_datum IN troskovi.datum%TYPE,
  p_naziv_usluge IN troskovi.naziv_usluge%TYPE,
  p_kolicina IN troskovi.kolicina%TYPE,
  p_br_rezervacije IN troskovi.br_rezervacije%TYPE
)
AS
  p_trosak_id varchar(20);
  p_count number;
  p_count_rez number;
BEGIN
  SELECT COUNT(*)
  INTO p_count
  FROM ponuda_cjenik
  WHERE naziv_usluge = p_naziv_usluge;

  SELECT COUNT(*)
  INTO p_count_rez
  FROM rezervacija
  WHERE br_rezervacije = p_br_rezervacije AND prijava <= p_datum AND odjava >= p_datum;

  IF p_count = 1 AND p_count_rez = 1 THEN
    SELECT CONCAT('T', brojevi_t.nextval)
    INTO p_trosak_id
    FROM DUAL;

    INSERT INTO troskovi(trosak_id, datum, naziv_usluge, kolicina, br_rezervacije)
    VALUES(p_trosak_id, p_datum, p_naziv_usluge, p_kolicina, p_br_rezervacije);
    COMMIT;
  ELSE
    DBMS_OUTPUT.PUT_LINE('Niste unijeli dobar naziv usluge ili dobar datum');
  END IF;
END unos_troskovi;
/

```

Procedura izdavanje_racuna

Procedura *izdavanje_racuna* je procedura koja pomoću broja rezervacije računa kolika je ukupna cijena računa za neku rezervaciju i također u tablicu *troskovi* dodaje troškove koje možemo vidjeti iz rezervacije. U kampovima se posebno naplaćuje boravak odrasle osobe, boravak djece, najam parcele

čija cijena ovisi o vrsti, boravišna pristojba i boravak kućnih ljubimaca, pa se ti troškovi ne unose ručno u tablicu troškovi, već ih procedura izdavanje_racuna unosi u tablicu troškovi prilikom izdavanja računa.

Za pokretanje procedure *izdavanje_racuna* potrebno je samo unijeti broj rezervacije. A procedura će sama provjeriti koliko je bilo odraslih, djece, kućnih ljubimaca, provjeriti vrstu parcele i sve te troškove unijeti u tablicu *troškovi* i nakon toga izdati račun, koji ima svoj jedinstveni broj, tj unijeti ga u tablicu *racuni* i status će mu biti 'izdan' te ga je nakon toga još potrebno platiti.

```
CREATE PROCEDURE izdavanje_racuna(
    p_br rezervacije IN rezervacija.br_rezervacije%TYPE
)
AS
    p_trosak number(7,2);
    p_count_rez number;
    br_odraslih number;
    br_djece number;
    p_datum_prijave date;
    p_br_dana number;
    p_br_ljubimci number;
    p_parcela varchar(20);
    p_count_ra number;
BEGIN
    SELECT COUNT(*)
    INTO p_count_rez
    FROM rezervacija
    WHERE br_rezervacije = p_br_rezervacije;

    SELECT COUNT(*)
    INTO p_count_ra
    FROM racuni
    WHERE br_rezervacije = p_br_rezervacije;

    IF p_count_rez=1 AND p_count_ra = 0 THEN
        SELECT prijava
        INTO p_datum_prijave
        FROM rezervacija
        WHERE br_rezervacije = p_br_rezervacije;

        SELECT odjava-prijava
        INTO p_br_dana
        FROM rezervacija
        WHERE br_rezervacije = p_br_rezervacije;

        SELECT COUNT(*)
        INTO p_br_ljubimci
        FROM rezervacija
        WHERE br_rezervacije = p_br_rezervacije;
```



```

SELECT vrsta_parcele
INTO p_parcela
FROM parcela pa
JOIN rezervacija re USING(br_parcele)
WHERE br_rezervacije = p_br_rezervacije;

SELECT COUNT(*)
INTO br_odraslih
FROM prijava
WHERE br_rezervacije = p_br_rezervacije AND
(SELECT ROUND(dob, 0)
FROM (SELECT TRUNC(MONTHS_BETWEEN(SYSDATE, datum_rodenja))/12 as dob
FROM DUAL)) > 18;

SELECT COUNT(*)
INTO br_djece
FROM prijava
WHERE br_rezervacije = p_br_rezervacije AND
(SELECT ROUND(dob, 0)
FROM (SELECT TRUNC(MONTHS_BETWEEN(SYSDATE, datum_rodenja))/12 as dob
FROM DUAL)) < 18 AND (SELECT ROUND(dob, 0)
FROM (SELECT TRUNC(MONTHS_BETWEEN(SYSDATE, datum_rodenja))/12 as dob
FROM DUAL))>2;

IF br_odraslih > 0 THEN
INSERT INTO troskovi(trosak_id, datum, naziv_usluge, kolicina, br_rezervacije)
VALUES(CONCAT('TT', brojevi_tt.nextval),p_datum_prijave, 'odrasli nocenje', br_odraslih*p_br_dana, p_br_rezervacije);
INSERT INTO troskovi(trosak_id, datum, naziv_usluge, kolicina, br_rezervacije)
VALUES(CONCAT('TT', brojevi_tt.nextval),p_datum_prijave, 'pristojba odrasli', br_odraslih*p_br_dana, p_br_rezervacije);
END IF;

IF br_djece > 0 THEN
INSERT INTO troskovi(trosak_id, datum, naziv_usluge, kolicina, br_rezervacije)
VALUES(CONCAT('TT', brojevi_tt.nextval),p_datum_prijave, 'djeca nocenje', br_djece*p_br_dana, p_br_rezervacije);
INSERT INTO troskovi(trosak_id, datum, naziv_usluge, kolicina, br_rezervacije)
VALUES(CONCAT('TT', brojevi_tt.nextval),p_datum_prijave, 'pristojba djeca', br_djece*p_br_dana, p_br_rezervacije);
END IF;

IF p_br_ljubimci > 0 THEN
INSERT INTO troskovi(trosak_id, datum, naziv_usluge, kolicina, br_rezervacije)
VALUES(CONCAT('TT', brojevi_tt.nextval),p_datum_prijave, 'kucni ljubimac', p_br_ljubimci*p_br_dana, p_br_rezervacije);
END IF;

INSERT INTO troskovi(trosak_id, datum, naziv_usluge, kolicina, br_rezervacije)
VALUES(CONCAT('TT', brojevi_tt.nextval),p_datum_prijave, p_parcela , p_br_dana, p_br_rezervacije);
COMMIT;

```

```

IF EXTRACT(MONTH FROM p_datum_prijave)>=6 AND EXTRACT(MONTH FROM p_datum_prijave)<=8 THEN
SELECT SUM(cijena)
INTO p_trosak
FROM (
SELECT kolicina*pc.cijena_sezona as cijena
FROM troskovi
JOIN ponuda_cjenik pc USING(naziv_usluge)
WHERE naziv_usluge=naziv_usluge AND br_rezervacije=p_br_rezervacije);

ELSE
SELECT SUM(cijena)
INTO p_trosak
FROM (
SELECT kolicina*pc.cijena_van_sezone as cijena
FROM troskovi
JOIN ponuda_cjenik pc USING(naziv_usluge)
WHERE naziv_usluge=naziv_usluge AND br_rezervacije=p_br_rezervacije);
END IF;
INSERT INTO racuni(br_racuna, br_rezervacije,iznos, datum, status)
VALUES(CONCAT('RA', brojevi_ra.nextval), p_br_rezervacije, p_trosak, p_datum_prijave + p_br_dana, 'izdan');
COMMIT;

ELSIF p_count_rez = 0 THEN
DBMS_OUTPUT.PUT_LINE('Niste unijeli dobar broj rezervacije.');
```

```

ELSE
DBMS_OUTPUT.PUT_LINE('Vec postoji racun za tu rezervaciju.');
```

```

END IF;
END izdavanje_racuna;
/

```

Ukupan iznos računa se računa tako da procedura pronađe sve troškove vezane za tu rezervaciju i pomnoži ih sa cijenom u sezoni ili cijenom izvan sezone ovisno o tome o kojem datumu rezervacije se radi.

Pozvat ćemo procedure *izdavanje_racuna* na rezervaciju Dore Doric tj rezervaciji broj 'R10040'.

```
CALL izdavanje_racuna('R10040');
```

Kako bi saznali broj računa i kolika je njegova cijena pozvat ćemo sljedeći upit:

```
SELECT *  
FROM racuni  
WHERE br_rezervacije = 'R10040';
```

Script Output	Query Result			
All Rows Fetched: 1 in 0,037 seconds				
BR_RACUNA	BR_REZERVACIJE	IZNOS	DATUM	STATUS
1 RA120	R10040	3099	20.07.20	izdan

Procedura *placanje_racuna*

Proceduru *placanje_racuna* koristimo nakon procedure *izdavanje_racuna*. Neki gosti znaju većer prije zatražiti izdavanje računa kako bi pripremili novce i zato procedura *izdavanje_racuna* status računa automatski stavlja na 'izdan', pa je potrebno pozvati porceduru *placanje_racuna* i unijeti broj računa te datum plaćanja. Broj računa uvijek počinje sa 'RA', a slično je i sa ostalim argumetima koji su primarni ključevi. Broj rezervacije 'R', ID unajmljivača 'U', ID troška 'T' ako je unesen ručno, a ako je unesen procedurom *izdavanje_racuna* 'TT', ID gosta 'G'.

```
CALL placanje_racuna('RA120', '20-07-2020');

SELECT *
FROM racuni
WHERE br_rezervacije = 'R10040';
```

Script Output x

Query Result x

BR_RACUNA	BR_REZERVACIJE	IZNOS	DATUM	STATUS
1 RA120	R10040	3099	20.07.20	placen

```

CREATE PROCEDURE placanje_racuna(
    p_br_racuna IN racuni.br_racuna%TYPE,
    p_datum IN date
)
AS
    p_count number;
BEGIN
    SELECT COUNT(*)
    INTO p_count
    FROM racuni
    WHERE br_racuna = p_br_racuna AND status = 'izdan';

    IF p_count = 1 THEN
        UPDATE racuni
        SET status = 'placen', datum = p_datum
        WHERE br_racuna = p_br_racuna;
    ELSE
        DBMS_OUTPUT.PUT_LINE('Racun s tim brojem ne postoji ili je vec placen.');

```

Upiti

Ako želimo saznati koje sve troškove nam račun sadržava, možemo pozvati sljedeći upit uz pomoć broja računa. Ako se radi o rezervaciji koja nije u razdoblju od 1.6. do 31.8. potrebno je *cijena sezone* promijeniti u *cijena_van_sezone*. Količina za odrasli nocenje je 22, jer imamo dvoje odraslih koji su u kampu boravili 11 noći...

-----upit za ispis svih troskova za dani broj racuna-----

```

SELECT datum, naziv_usluge, kolicina, pc.cijena_sezona, kolicina*pc.cijena_sezona as ukupno
FROM troskovi t
JOIN ponuda_cjenik pc USING(naziv_usluge)
WHERE naziv_usluge=naziv_usluge AND br_rezervacije IN (SELECT br_rezervacije
FROM racuni
WHERE br_racuna = 'RA120'
)
ORDER BY datum;

```

Script Output x Query Result x

SQL | All Rows Fetched: 11 in 0,043 seconds

DATUM	NAZIV_USLUGE	KOLICINA	CIJENA_SEZONA	UKUPNO
1 09.07.20	djeca nocenje	11	34	374
2 09.07.20	odrasli nocenje	22	57	1254
3 09.07.20	sator	11	46	506
4 09.07.20	kucni ljubimac	11	23	253
5 09.07.20	pristojba odrasli	22	8	176
6 09.07.20	pristojba djeca	11	4	44
7 09.07.20	parking auto	11	31	341
8 10.07.20	lezaljka	1	35	35
9 14.07.20	rostitlj	1	25	25
10 15.07.20	masina	1	31	31
11 18.07.20	pedalina sat	1	60	60

Ako želimo saznati sve troškove uz pomoć broja rezervacije, možemo koristiti sljedeći upit:

-----svi troskovi za danu rezervaciju-----

```

SELECT datum, naziv_usluge, kolicina, pc.cijena_sezona, kolicina*pc.cijena_sezona as ukupno
FROM troskovi t
JOIN ponuda_cjenik pc USING(naziv_usluge)
WHERE naziv_usluge=naziv_usluge AND br_rezervacije='R10040'
ORDER BY datum;

```

Script Output x Query Result x

SQL | All Rows Fetched: 11 in 0,045 seconds

DATUM	NAZIV_USLUGE	KOLICINA	CIJENA_SEZONA	UKUPNO
1 09.07.20	djeca nocenje	11	34	374
2 09.07.20	odrasli nocenje	22	57	1254
3 09.07.20	sator	11	46	506
4 09.07.20	kucni ljubimac	11	23	253
5 09.07.20	pristojba odrasli	22	8	176
6 09.07.20	pristojba djeca	11	4	44
7 09.07.20	parking auto	11	31	341
8 10.07.20	lezaljka	1	35	35
9 14.07.20	rostitlj	1	25	25
10 15.07.20	masina	1	31	31
11 18.07.20	pedalina sat	1	60	60

Ako želimo ispisati sve goste koji se na dani datum nalaze u kampu i stariji su od 18 godina koristit ćemo sljedeći upit:

```

-----ispis svih gostiju starijih od 18 godina za dani datum-----
SELECT gost_id, ime, prezime, broj_dokumenta, datum_rodenja, ROUND(TRUNC(MONTHS_BETWEEN(SYSDATE, datum_rodenja))/12,0) as dob, br_rezervacije
FROM prijava
WHERE br_rezervacije IN (SELECT br_rezervacije
FROM rezervacija
WHERE prijava <= TO_DATE('09-07-2020') AND odjava >= TO_DATE('09-07-2020'))
AND (SELECT ROUND(dob, 0)
FROM (SELECT TRUNC(MONTHS_BETWEEN(SYSDATE, datum_rodenja))/12 as dob
FROM DUAL)) > 18;

```

GOST_ID	IME	PREZIME	BROJ_DOKUMENTA	DATUM_RODENJA	DOB	BR_REZERVACIJE
1 G123	Kristijan	Martinov	32577247	13.12.96	24	R10004
2 G124	Lena	Mihic	13592316	17.05.97	23	R10004
3 G125	Matija	Vuko	47838788	18.10.99	21	R10004
4 G126	Sebastijan	Grgic	3748290	23.10.97	23	R10004
5 G140	Iva	Linic	7439291	08.11.87	33	R10013
6 G141	Dorotea	Baric	7439291	09.08.89	31	R10013
7 G142	Marina	Dedic	1357351	28.01.89	31	R10013
8 G143	Ana	Horvat	58829991	29.11.85	35	R10012
9 G144	Darko	Bradaric	6962892	07.02.82	38	R10012
10 G147	Nikola	Zubcic	4682781	25.04.90	30	R10011
11 G148	Marina	Zubcic	8520038	12.09.92	28	R10011
12 G149	Luka	Konjevic	8390012	30.12.89	31	R10011
13 G150	Rebeka	Malinovic	24749291	25.04.90	30	R10011
14 G151	Ivana	Dasovic	8245294	25.03.79	41	R10015
15 G152	Goran	Dasovic	3742181	12.09.81	39	R10015
16 G154	Sinisa	Lukic	4238902	21.07.78	42	R10015
17 G155	Dubravka	Lukic	2429084	07.11.80	40	R10015
18 G162	Dora	Doric	4352789	14.08.93	27	R10040
19 G163	Marko	Doric	8352368	19.03.92	28	R10040

Da bi saznali koliko je gostiju u danom trenutku u kampu, pozvat ćemo ovaj upit:

```

-----broj prijavljenih gostiju za dani datum-----
SELECT COUNT(*) as broj_gostiju
FROM prijava
WHERE br_rezervacije IN (SELECT br_rezervacije
FROM rezervacija
WHERE prijava <= TO_DATE('09-07-2020') AND odjava >= TO_DATE('09-07-2020'));

```

BROJ_GOSTIJU
24

Sve dostupne parstele za dano razdoblje možemo saznati uz pomoć ovog upita:

```
-----izbacuje sve parcele dostupne za zadano razdoblje-----
SELECT * FROM parcela pa
WHERE br_parcele NOT IN
  (SELECT br_parcele FROM rezervacija
   WHERE br_parcele = pa.br_parcele AND TO_DATE('09-07-2020', 'DD-MM-YYYY') < odjava AND TO_DATE('20-07-2020', 'DD-MM-YYYY') > prijava);
```

Script Output x Query Result x

All Rows Fetched: 1 in 0,062 seconds

BR_PARCELE	VRSTA_PARCELE	OPIS_PARCELE
1	5 sator	Parcela u hladu za sator.

Ako nas zanima koliko je zauzetih parcela s obzirom na vrstu za dani datum koristit ćemo sljedeći upit:

```
-----broj zauzetih parcela po vrsti za dani datum-----
SELECT SUM(CASE WHEN vrsta_parcele = 'sator' THEN 1 ELSE 0 END) br_satora,
       SUM(CASE WHEN vrsta_parcele = 'kamp kucica' THEN 1 ELSE 0 END) br_kamp_kucica,
       SUM(CASE WHEN vrsta_parcele = 'kamp prikolica' THEN 1 ELSE 0 END) br_kamp_prikolica
FROM parcela
JOIN rezervacija USING(br_parcele)
WHERE br_parcele = br_parcele AND prijava <= TO_DATE('09-07-2020') AND odjava >= TO_DATE('09-07-2020');
```

Script Output x Query Result x

All Rows Fetched: 1 in 0,044 seconds

BR_SATORA	BR_KAMP_KUCICA	BR_KAMP_PRIKOLICA
1	2	3

Indexi

```
CREATE INDEX i_br_rezervacije
ON troskovi(br_rezervacije);
```

Index *i_br_rezervacije* je index na tablicu *troskovi* na argument *br_rezervacije*. Ovaj index utječe na upit za ispis svih troškova vezanih za neku rezervaciju i na upit za ispis svih troškova vezanih za broj računa.

Prije kreiranja indexa, potrebno vrijeme za izvršavanje upita bilo je 0,04 sekunde, dok je nakon kreiranja upita potrebno vrijeme bilo 0,029 sekundi.

-----svi troskovi za danu rezervaciju-----

```

SELECT datum, naziv_usluge, kolicina, pc.cijena_sezona, kolicina*pc.cijena_sezona as ukupno
FROM troskovi t
JOIN ponuda_cjenik pc USING(naziv_usluge)
WHERE naziv_usluge=naziv_usluge AND br_rezervacije='R10040'
ORDER BY datum;

```

Script Output x Query Result x

SQL | All Rows Fetched: 11 | 0,04 seconds

	DATUM	NAZIV_USLUGE	KOLICINA	CIJENA_SEZONA	UKUPNO
1	09.07.20	djeca nocenje	11	34	374
2	09.07.20	odrasli nocenje	22	57	1254
3	09.07.20	sator	11	46	506
4	09.07.20	kucni ljubimac	11	23	253
5	09.07.20	pristojba odrasli	22	8	176
6	09.07.20	pristojba djeca	11	4	44
7	09.07.20	parking auto	11	31	341
8	10.07.20	lezaljka	1	35	35
9	14.07.20	rostitlj	1	25	25
10	15.07.20	masina	1	31	31
11	18.07.20	pedalina sat	1	60	60

Prije kreiranja indexa

-----svi troskovi za danu rezervaciju-----

```

SELECT datum, naziv_usluge, kolicina, pc.cijena_sezona, kolicina*pc.cijena_sezona as ukupno
FROM troskovi t
JOIN ponuda_cjenik pc USING(naziv_usluge)
WHERE naziv_usluge=naziv_usluge AND br_rezervacije='R10040'
ORDER BY datum;

```

Script Output x Query Result x

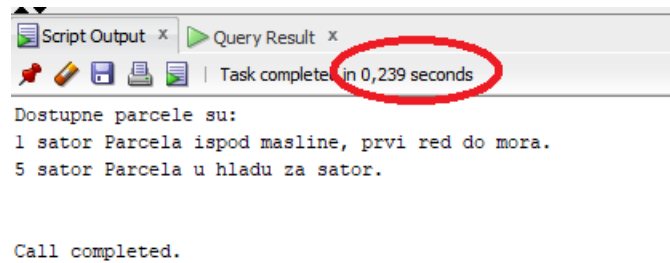
SQL | All Rows Fetched: 11 | 0,029 seconds

	DATUM	NAZIV_USLUGE	KOLICINA	CIJENA_SEZONA	UKUPNO
1	09.07.20	djeca nocenje	11	34	374
2	09.07.20	odrasli nocenje	22	57	1254
3	09.07.20	sator	11	46	506
4	09.07.20	kucni ljubimac	11	23	253
5	09.07.20	pristojba odrasli	22	8	176
6	09.07.20	pristojba djeca	11	4	44
7	09.07.20	parking auto	11	31	341
8	10.07.20	lezaljka	1	35	35
9	14.07.20	rostitlj	1	25	25
10	15.07.20	masina	1	31	31
11	18.07.20	pedalina sat	1	60	60

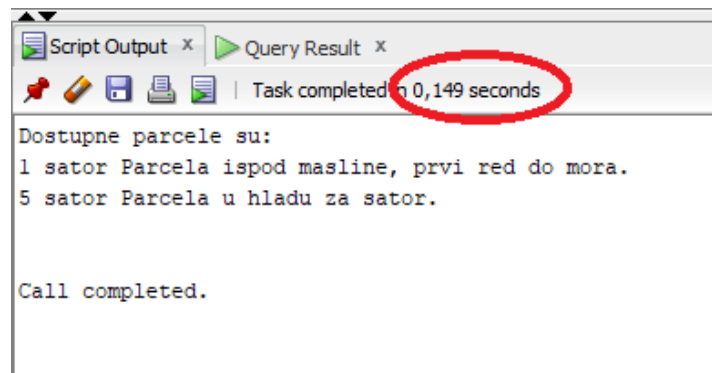
Nakon kreiranja indexa

```
CREATE INDEX i_datum  
ON rezervacija(prijava, odjava);
```

Index *i_datum* je index nad tablicom *rezervacija* nad argumentima *prijava* i *odjava*. Ovaj index je kreiran kako bi se ubrzala procedura *dostupnost_parcela*.



Prije kreiranja indexa



Nakon kreiranja indexa